Hasso Plattner Institut

IT Systems Engineering | Universität Potsdam

Der Bootprozess unter Linux

Betriebssystemadministration | David Jaeger | 21. Mai 2008

Inhalt

- Einführung
- Grober Überblick zum Bootprozess
- Startmodelle für UNIX-Systeme
- System-V-Runlevel
- Der Init-Prozess und seine Konfigurationsdatei inittab
- Startscripte und deren Speicherort
- Dienste unter Linux
- Demonstration eines Dienstes
- Zusammenfassung

Einführung

- Linux für Serversysteme weit verbreitet
- Bootprozess sollte gut verstanden werden
  - Was sollte man wissen?
    - Speicherorte von Konfigurationdateien
    - Speicherorte von Diensten
    - Grundlegendes Verständnis für Startprozess
  - Warum sollte man den Bootprozess gut verstehen?
    - Korrektur von Abhängigkeiten in Startreihenfolge von Diensten
    - Entfernen von Diensten aus dem Startvorgang
    - Richtige Einordnung eigener Dienste in Startvorgang
    - Schreiben von eigenen Initscripten

Grober Überblick zum Bootprozess

- Alles beginnt im BIOS
- Stages – Die Etappen zum Erfolg
- Der Kernel – Kern von Linux

Alles beginnt im BIOS

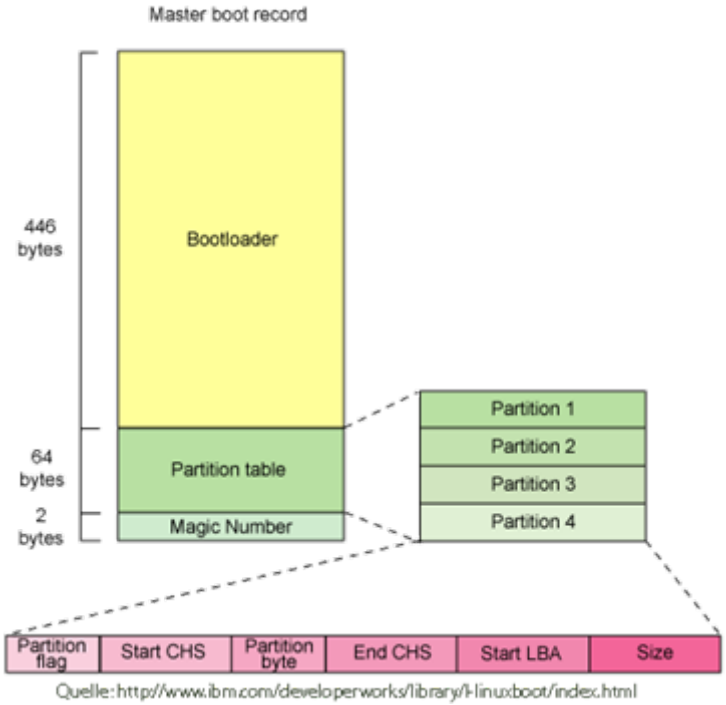
- Prozessor führt erste Anweisungen im BIOS aus
  - **P**ower-**O**n **S**elf **T**est
- Bestimmen der bootfähigen Medien
- Suchen nach Bootsektoren auf Geräten
  - Ausführen des Codes im Bootsektor

Stages – Die Etappen zum Erfolg (1/2)

- Laden des Linux-Kernels in Stages unterteilt

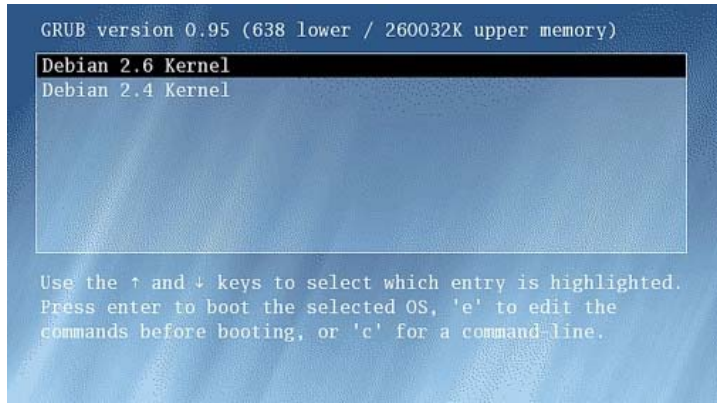
Stage 1 Boot Loader - MBR

- Liegt im Master Boot Record (MBR) des bootfähigen Geräts
- Größe: 512 Byte
- Aufgaben:
  - Suche nach aktiver Partition im MBR
  - Laden von Stage 2 Boot Loader aus Boot Record der aktiven Partition



Stage 2 Boot Loader – Laden des Kernelimages

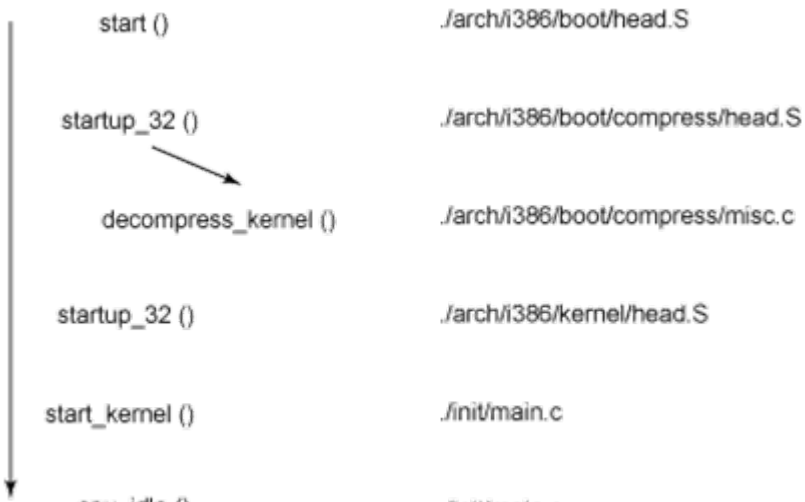
- Liegt im Boot Record der aktiven Partition
- Mit *Stage 1 Boot Loader* in *LILO* oder *GRUB* implementiert
- Aufgaben:
  - Ggf. Anzeige des Boot-Splashs mit Auswahl des zu ladenden Kernelimages (GRUB/LILO)
  - Lädt das Linux-Kernelimage in den RAM
  - Lädt optional *initial RAM disk* (initrd)



Der Kernel – Kern unter Linux (1/3)

- Kernelimage komprimiert in *zImage*/*bzImage*

1. Dekompressionsfunktion am Anfang des Images wird aufgerufen
2. Funktion entpackt Kernel in oberen RAM-Bereich
3. Starten des eigentlichen Kernels



Quelle: http://www.ibm.com/developerworks/library/l-linuxboot/index.html

Der Kernel – Kern unter Linux (2/3)

start: *(./arch/i386/boot/head.S)*

- Initialisierung einiger Hardware

startup_32: *(./arch/i386/boot/compress/head.S)*

- Aufbauen der Ausführungsumgebung (Stack, etc.)

decompress_kernel: *(./arch/i386/boot/compress/misc.c)*

- Entpacken des Kernelimages

Der Kernel – Kern unter Linux (3/3)

startup_32: *(./arch/i386/kernel/head.S)*

- Auch Prozess 0 oder Swapper genannt
- Initialisierung von Seitentabellen und Speichermapping
- Feststellen des CPU-Typs

start_kernel: (./init/main.c)

- Eintrittspunkt für den Kernel
- Ruft sämtliche Initialisierungfunktionen auf
- Laden der *initial RAM disk* (initrd)

kernel_thread:

- Startet Initialisierungprozess

Startmodelle für UNIX-Systeme

BSD-Modell:

- Beschränkte Anzahl von Startscripten
- Basisscripte: /etc/rc, /etc/rc.boot, /etc/rc.local
- Benutzt in BSD-Systemen

System-V Modell:

- Komplexer Satz von Startscripten
- Basiert auf Prinzip von Runleveln (Systemzuständen)
- Ladezeitpunkt von Programmen durch Runlevel bestimmt
- Runlevel wird Verzeichnis zugeordnet
- Startscripte für Programme liegen in Verzeichnissen
- Benutzt in Linux-Systemen

System-V-Runlevel

Runlevel –
Die Statuslevel eines Linux

Was versteht man unter Runleveln?

- Bestandteil des System-V-Startvorgangs
- Signalisieren Zustand des Systems
- Komplett durch Software bereitgestellt
  - Implementiert durch init mit Konfiguration inittab
  - Runlevel als Parameter für init-Prozess
- Üblicherweise 7 verschiedene Runlevel (bis zu 10)
  - Zuordnung bei Distributionen unterschiedlich
  - Möglichkeit Runlevel selbst zu definieren (z.B. Level 4)

Runlevel – Standard-Runlevel

Runlevel 0:

- Beendet alle laufenden Prozesse und hält das System an

Runlevel S: (Häufig als Runlevel 1)

- Einzelnutzermodus
- Ohne Netzwerk
- Keine Dienste
- Wird für Wartungsarbeiten benutzt

Runlevel 6:

- Beendet alle laufenden Prozesse und startet das System neu

Runlevel –
Typische Runlevel für Linux

Runlevel 2:

- Mehrbenutzermodus
- Ohne Netzwerkunterstützung
- Dienste werden nicht geladen

Runlevel 3:

- Mehrbenutzermodus
- Mit Netzwerkunterstützung und allen Diensten

Runlevel 5:

- Wie Runlevel 3, aber mit X-Window-System

Runlevel – Besonderheiten einiger Distributionen

- Runlevel 4 ist meistens nicht belegt

Debian:

- Runlevel 2 – 5 identisch
  - Unterstützen Netzwerk mit Displaymanager

Slackware Linux:

- Level 4 für Mehrbenutzermodus
- Level 5 nicht belegt

Gentoo Linux:

- Level 4 und 5 sind identisch mit Level 3

Der Init-Prozess und seine Konfigurationsdatei inittab

Der Init-Prozess

- Ausführung als letzer Schritt beim Bootprozess
- Erster Usermode-Prozess (PID immer 1)
- Durch /sbin/init repräsentiert
  - Konfiguration durch /etc/inittab
- Laden/Beenden sämtlicher Dienste/Programme während bestimmter Systemstadien
- Prozess wird niemals beendet

/etc/inittab – Allgemeines

- Konfigurationdatei für init-Prozess
- Abbildung von Runlevel auf Startscripte

Welche Sachverhalte definiert die inittab?

- Script, mit dem ein System gestartet wird
- Runlevel, in welches das System gebootet wird
- Script, das bei einem Runlevel gestartet wird
- Aktion bei Ctrl-Alt-Del

/etc/inittab – Aufbau

Jede Zeile mit vorgeschriebenem Format:

label:runlevel:aktion:prozess

label: beliebige Zeichenkette (meistens Länge 2)

runlevel: Nummer des zugeordneten Runlevels

aktion: z.B. wait, sysinit

prozess: Prozess, der für das Runlevel gestartet werden soll

/etc/inittab – Beispiel

Ausschnitt aus einer /etc/inittab:

```
si::sysinit:/etc/rc.d/rc.sysinit

l0:0:wait:/etc/rc.d/rc 0
l1:1:wait:/etc/rc.d/rc 1
l2:2:wait:/etc/rc.d/rc 2
l3:3:wait:/etc/rc.d/rc 3
l4:4:wait:/etc/rc.d/rc 4
l5:5:wait:/etc/rc.d/rc 5
l6:6:wait:/etc/rc.d/rc 6
```

- wait als Standardaktion für Runlevel
- /etc/rc.d/rc wird als Standardstartsscript benutzt

/etc/rc.d/rc

- Standardscript für Umsetzung von Runlevels
- Benötigt Runlevel als Parameter
- Startet Scripte für angegebenes Runlevel
- Führt Inhalt aus Verzeichnis /etc/rcN.d aus
  - N beschreibt Runlevel

Startscripte und deren Speicherort

/etc/rcN.d/ (1/2)

rcN.d beinhaltet Dateien, die für die Ausführung von Programmen während des Runlevels N sorgen.

Inhalt der Verzeichnisse:

- Symbolische Links auf Initscripte in /etc/init.d

Format der Dateien:

- [K|S][00-99]Programmname
  - **[K|S]:** K = Kill, S = Start
  - **[00-99]:** Beliebige natürliche Zahl
  - **Programmname:** Programmname in /etc/init.d

/etc/rcN.d/ (2/2)

Welche Aktion wird durchgeführt?

Das Script, auf den der Link verweist, wird…

- mit S beim Eintritt ins Runlevel gestartet.
- mit K beim Austritt aus dem Runlevel gestoppt.

In welcher Reihenfolge werden die symbolischen Links ausgeführt?

- Gilt sowohl für S als auch für K
- Natürliche Zahl als primäre Reihenfolge
- Bei gleicher Zahl entscheidet lexikographische Reihenfolge der Programmnamen

rc.local

- Teil des BSD-Startmodells
  - Einfach durch Benutzer verwendbar
- Lokales Konfigurationsscript
- inittab oder Initscripts sollten nicht direkt verändert werden
- Befindet sich in /etc/init.d
- Symbolischer Link mit S99local
  - Startet als letztes Script beim Booten

Initscripte

- Dienen als Start-/Stopscripte
- Befinden sich in /etc/init.d
  - Befinden sich an zentraler Stelle
  - Aussagekräftige Namen

Aufbau:

- Normale Shellscripts
- Sollte Aktionen für *start*, *stop*, *restart*, *status* implementieren

Dienste unter Linux

Was versteht man unter Diensten?

- Dienste unter Linux auch Daemonen genannt
- Meistens erkennbar an Programmendung d
- Prozesse, die ständig verfügbar sein müssen
- Laufen im Hintergrund
  - Input/Output nicht über Shell möglich
- Werden mit Hilfe von Initscripten angesprochen
- Kann zu bestimmen Systemstadien gestartet werden
  - Verwaltung mit /etc/init

Daemonisierung eines Prozesses (1/2)

- Jeder Prozess kann zu Daemon werden
- Umwandlung in Binary selbst oder über Initscript

Schritte zur Erstellung eines Daemons:

- Umstellung des ausführenden Nutzers
  - Zugriffseinschränkungen für den Dienst
- Erstelle Kindprozess
  - Aufruf für Service soll sofort zurückkehren
- Ignoriere sämtliche Nachrichten von außen
  - Services laufen im Hintergrund

Daemonisierung eines Prozesses (2/2)

- Maskiere Zugriff
  - Setze Standardzugriff für erstellte Dateien
- Ändere die Prozessgruppe
  - Unabhängigkeit von zusammenhängenden Prozessen
- Setze das Arbeitsverzeichnis zum Wurzelverzeichnis /
  - Vermeide Fehler durch Entfernen von Pfaden (umount)
- Schließe die Standard-Dateihandle (IN, OUT, ERR)
  - Ein Service kommuniziert unabhängig von der Shell
- Erstellen einer PID-Datei
  - Ausschluss des mehrmaligen Startens des Services

Typische Startreihenfolge einiger Dienste

/etc/rcS.d/:

1. Mounten der lokalen Laufwerke
2. Konfiguration der Netzwerkgeräte
3. Mounten von Netzlaufwerken
4. Starten von Soundtreibern
5. Initialisierung des Zufallspools
6. Laden des grafischen Frontends

/etc/rc2.d/:

1. Starten der Logginginfrastruktur (syslogd)
2. Starten sämtlicher Netzwerkdienste

Demonstration eines Dienstes

Zusammenfassung

- Laden des Kernelimages und entpacken des Kernels
- Initialisierung des Systems
- Init-Prozess letzter Schritt beim Bootprozess
  - Konfiguration durch inittab
  - Verwaltung von Initscripten in Runleveln
- Initscripte liegen in Verzeichnissen /etc/rcN.d
  - N enspricht Runlevel
  - Ausführungsreihenfolge nach lexikographischer Reihenfolge
  - Enthält nur Links auf Initscripte
- Eigentliche Initscripte liegen in /etc/init.d
- Initscripte starten Daemonen

Quellen

[1] - M. Tim Jones: “Inside the Linux boot process”,
http://www.ibm.com/developerworks/library/l-linuxboot/index.html

[2] - Craig Hunt: TCP/IP Netzwerk-Administration, O’Reilly

[3] - Greg Ippolito: Linux Init Process / PC Boot Procedure
http://www.yolinux.com/TUTORIALS/LinuxTutorialInitProcess.html

[4] - http://en.wikipedia.org/wiki/Runlevel

[5] - Doug Potter: “How to Daemonize in Linux”,
http://www-theorie.physik.unizh.ch/~dpotter/howto/daemonize

Danke für die Aufmerksamkeit